

**IF STATEMENT HAVING AN EXPRESSION SETUP CLAUSE TO BE
UTILIZED IN STRUCTURED ASSEMBLY LANGUAGE PROGRAMMING**

5

BACKGROUND OF THE INVENTION

1. Technical Field

The present invention relates to assembly programming in general, and in particular to structured assembly language programming. Still more particularly, the present invention relates to an IF construct having an expression setup clause to be utilized in structured assembly language programming.

2. Description of the Prior Art

Structured assembly language programming is an improvement to the basic syntax of conventional assembly language programming. In essence, structured assembly language programming allows the use of structured programming constructs similar to those generally found in high-level programming languages such as Pascal or C.

One of the structured programming constructs is the well-known IF construct. It consists of an IF clause followed by zero or more ELSE_IF clauses, an optional ELSE clause, and an END_IF statement. The simplest structured assembly language IF construct is typically utilized as follows:

```

condition setup assembly language code
IF (condition)
    conditionally executed assembly language code
END_IF
unassociated with IF assembly language code

```

where condition contains structured assembly language expression(s) for generating comparison opcodes (*i.e.*, opcodes that set the processor's flags) and/or branch opcodes. Only comparison opcodes and branch opcodes are used in the implementation of a structured assembly language expression. An assembler then converts the above-mentioned structured assembly language IF construct to a group of processor opcodes as follows (in assembler mnemonics):

```

condition setup assembly language code
cmp r0,#7                ; (r0 == 7) || (r1 < r2)
bnz falselabel
cmp r1,r2
bge falselabel
                        conditionally executed assembly language code
falselabel equ $

```

In this example, the condition compares a processor register to a numeric value, and a comparison also occurs between two processor registers. The condition is $(r0 == 7)$ or $(r1 < r2)$. One important difference between a structured assembly language IF construct and an IF construct from a high-level programming language is that a programmer is responsible for register setup so the structured assembly language expression, that is the condition, is only made up of comparison opcodes and/or branch opcodes. In fact, such kind of optimization capability is a key reason for choosing assembly language in the first place. The need to execute opcodes to setup for a structured assembly language expression is a real problem when multiple conditions must be tested. In order to test many different conditions using structured assembly language under the prior art, a nested testing scheme is required, and the code will resemble the following:

10
15
20
25
30

```

condition setup code1
IF (condition1)
    conditionally executed code1
ELSE
    condition setup code2
    IF (condition2)
        conditionally executed code2
    ELSE
        condition setup code3
        IF (condition3)
            conditionally executed code3
        ELSE
            condition setup code4
            IF (condition4)
                conditionally executed code4
            ELSE
                condition setup code5
                IF (condition5)
                    conditionally executed code5a
                ELSE
                    conditionally executed code5b
                END_IF
            END_IF
        END_IF
    END_IF
END_IF

```

It becomes obvious that multiple testing conditions are awkward to code in structured assembly language when standard programming conventions for indentation are followed. The awkward form of coding also makes the task of understanding code and the task of maintaining code more difficult. Consequently, it would be desirable to provide an improved IF statement to be utilized in structured assembly language programming for coding multiple testing conditions.

SUMMARY OF THE INVENTION

In accordance with a preferred embodiment of the present invention, a state machine for an assembler capable of processing structured assembly language IF constructs includes five states, namely, an IF state, an ELSE state, an END_IF state, an ELSE_IF state, and a SETUP_IF state. In response to recognizing a SETUP_IF clause during the IF state or the ELSE_IF state, the process transitions to the SETUP_IF state. In response to recognizing an ELSE_IF clause during the SETUP_IF state, the process transitions to the ELSE_IF state.

All objects, features, and advantages of the present invention will become apparent in the following detailed written description.

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198

BRIEF DESCRIPTION OF THE DRAWINGS

The invention itself, as well as a preferred mode of use, further objects, and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 is a state machine for the standard structured assembly language IF construct;

Figure 2 is a state machine for a structured assembly language IF construct in accordance with a preferred embodiment of the present invention;

Figure 3 is a block diagram of an assembler to which a preferred embodiment of the present invention can be applied; and

Figure 4 is a block diagram of a computer system to which a preferred embodiment of the present invention can be applied.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

A more specific example of nested testing, using a standard programming indentation convention, is shown below:

```

5      ldr r0,[r2+0]                                ;0
      IF (r0 <ne> #7)
          add r1,#1
10     ELSE
          ldr r0,[r2+3]                                ;1
          IF (r0 <ge> #3)
              add r1,#2
          ELSE
15              ldr r0,[r2+5]                                ;2
              IF (r0 <le> #1)
                  add r1,#3
              ELSE
20                  add r1,#4
                  END_IF
          END_IF
      END_IF

```

A careful examination of the above-shown nested testing reveals that each IF construct is inside an ELSE clause. Thus, the logic is really IF-ELSEIF-ELSEIF ... ENDIF. Unfortunately, the nested structured assembly language IF constructs hide this fact.

In accordance with a preferred embodiment of the present invention, a SETUP_IF clause is introduced to eliminate the confusion caused by the nested IFs, and to restore the logic to a form that truly represents what is happening. The modified IF construct using the SETUP_IF clause consists of an IF clause followed by an optional SETUP_IF clause preceding an ELSE_IF clause, an optional ELSE clause, and an END_IF statement. The pair of clauses, an optional SETUP_IF clause followed by an ELSE_IF clause, may appear zero or more times immediately following the IF clause, as follows:

```

condition setup code1
IF (condition1)
    conditionally executed code1
SETUP_IF          ;branch endif, label for "condition1 is false"
5   condition setup code2
ELSE_IF (condition2)
    conditionally executed code2
SETUP_IF          ;branch endif, label for "condition2 is false"
10  condition setup code3
ELSE_IF (condition3)
    conditionally executed code3
SETUP_IF          ;branch endif, label for "condition3 is false"
15  condition setup code4
ELSE_IF (condition4)
    conditionally executed code4
ELSE_IF (condition5)
    conditionally executed code5a
ELSE              ;branch endif, label for "condition5 is false"
    conditionally executed code5b
END_IF            ;label for endif, label for "condition6 is false"

```

The SETUP_IF statement, at the start of the SETUP_IF clause, is converted by an assembler to a branch to END_IF label. The SETUP_IF statement will also define the false label needed by the previous structured assembly language expression. If condition1 is true, code1 is then executed. The branch END_IF code generated by the SETUP_IF statement immediately following the IF clause is then executed. If condition1 is false, the code for condition1 will branch to the false label1 generated by the SETUP_IF statement immediately following the IF clause. Execution will continue with setup code2. The above-mentioned pattern is repeated for all instances of the SETUP_IF clause in the modified structured assembly language IF construct.

The above-mentioned multiple testing condition example can be coded using SETUP_IF clauses, as follows:

```

    ldr r0,[r2+0]                                ;0
    IF (r0 <ne> #7)
        add r1,#1
    SETUP_IF
5      ldr r0,[r2+3]                                ;1
    ELSE_IF (r0 <ge> #3)
        add r1,#2
    SETUP_IF
10     ldr r0,[r2+5]                                ;2
    ELSE_IF (r0 <le> #1)
        add r1,#3
    ELSE
        add r1,#4
    END_IF

```

Compared with the previous code, the new code does not have a requirement for the programmer to balance multiple IF with multiple END_IF statements. Thus, without the present invention, each condition tested requires the assembler to keep and to track an instance of an IF construct. But with the present invention, only a single IF construct is required. Because structured assembly language, like structured programming, is block-oriented, each new IF statement is a new unique block. Each structured assembly language block requires the assembler to track the state of the a block, to generate and remove the block and its related information, and to allocate and consume assembler resources. Fewer nested structured assembly language blocks means fewer resources are required by the assembler.

Generally speaking, structured assembly language is implemented through program labels, pattern generated code, a stack for structured programming blocks, and a simple state machine for each different type of structured assembly language construct. The structured assembly language IF construct is no exception. Thus, the present invention can also be compared with the prior art from a state machine point of view. The state machine for the standard structured assembly language IF construct is depicted in Figure 1. As shown, a state machine x has four states, namely, an IF state, an ELSE state, an END_IF state, and an ELSE_IF state. Each state represents a clause in an IF construct. Steps and actions for each state are summarized in Table I.

Table I

State	Description of Steps for state
IF	1) Push new block on structured assembly block stack 2) Generate a new and unique NextClauseLabel (false label) 3) Generate EndIfLabel 4) Emit opcode(s) for condition
ELSEIF	1) Emit jump to EndIfLabel 2) Set NextClauseLabel (next false label) 3) Generate a new and unique NextClauseLabel (false label) 4) Emit opcode(s) for condition
ELSE	1) Emit jump to EndIfLabel 2) Set NextClauseLabel (false label)
ENDIF	1) Set NextClauseLabel (false label) 2) Set EndIfLabel 3) Pop current block off structured assembly block stack

In accordance with a preferred embodiment of the present invention, the state machine of the present invention is depicted in Figure 2. As shown, a state machine y has five states, namely, an IF state, an ELSE state, an END_IF state, an ELSE_IF state, and a SETUP_IF state. Each state represents an IF construct clause. Steps and actions for each state is summarized in Table II.

5
10
15

Table II

State	Description of Steps for state	Status
IF	1) Push new block on structured assembly block stack 2) Generate NextClauseLabel (false label) 3) Generate EndIfLabel 4) Emit opcode(s) for condition	unchanged
SETUP	1) Emit jump to EndIfLabel 2) Set NextClauseLabel (false label) 3) Generate a new and unique Next ClauseLabel (false label)	new
ELSEIF	1) Emit jump to EndIfLabel if previous state was not setup 2) Set NextClauseLabel 3) Generate a new and unique NextClauseLabel (false label) 4) Emit opcode(s) for condition	modified
ELSE	1) Emit jump to EndIfLabel 2) Set NextClauseLabel	unchanged
ENDIF	1) Set NextClauseLabel 2) Set EndIfLabel 3) Pop current block off structured assembly block stack	unchanged

One of the virtues of structured assembly programming is that the internal structures representing constructs in structured assembly language can be stored in a stack. Each structured assembly language construct or block needs to be identified by a type, and the construct's internal state needs to be recorded, and two labels are required, one label for the next clause and one label for the end of the construct. The manipulation of those values is shown in Table I and Table II. As shown, the nature of the elements in a tuple that holds the state of the IF construct state machine remains unchanged.

The relationship between the state values of a structured assembly language construct and an assembler is shown in Figure 3, there is illustrated a block diagram of an assembler to which a preferred embodiment of the present invention can be applied. As shown, an assembler 30 includes a lexer 32, a parser 33, and a code generator 34. User source files 31 are broken down into tokens by lexer 32. Groups of tokens are recognized

5

An assembler, such as assembler 30, capable of processing structured assembly language may be executed in a variety of data processing systems under a number of different operating systems. The computer may be, for example, a personal computer, a midrange computer or a mainframe computer. In addition, the computer may be a stand-alone system or part of a network such as a local-area network (LAN) or a wide-area network (WAN).

With reference now to Figure 4, there is depicted a block diagram of a computer system 10 in which a preferred embodiment of the present invention is applicable. As shown, a processor 12, a read-only memory (ROM) 13, and a random access memory (RAM) 14 are connected to a system bus 11. Processor 12, ROM 13, and RAM 14 are also coupled to a peripheral component interconnect (PCI) bus 20 of computer system 10 through a PCI host bridge 16. PCI host bridge 16 provides a low latency path through which processor 12 may directly access PCI devices mapped anywhere within bus memory and/or I/O address spaces. PCI host bridge 16 also provides a high bandwidth path allowing PCI devices to directly access RAM 14.

In addition, an audio adapter **23** and a graphics adapter **21** may be attached to PCI bus **20**. Graphics adapter **21** controls visual output through a video monitor **22** and audio adapter **20** controls audio output through a speaker **24**. Also attached to PCI bus **20** is a communications adapter **15** and a small computer system interface (SCSI) **18**. Communications adapter **15** connects computer system **10** to a local-area network (LAN) **17**. SCSI **18** is utilized to control a high-speed SCSI disk drive **19**. Expansion bus bridge

29, such as a PCI-to-ISA bus bridge, may be utilized for coupling an industry standard architecture (ISA) bus 25 to PCI bus 20. As shown, a keyboard 26 and a mouse 28 are attached to ISA bus 25 for performing certain basic I/O functions.

5 As has been described, the present invention provides a SETUP_IF clause, which is an ELSE_IF clause having an expression setup clause, to be utilized in structured assembly language programming.

It is also important to note that although the present invention has been described in the context of a fully functional computer system, those skilled in the art will appreciate that the mechanisms of the present invention are capable of being distributed as a program product in a variety of forms, and that the present invention applies equally regardless of the particular type of signal bearing media utilized to actually carry out the distribution. Examples of signal bearing media include, without limitation, recordable type media such as floppy disks or CD ROMs and transmission type media such as analog or digital communications links.

20 While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.